Edge detection:

image * filter = output

↑ convolution operation

Vertical edge : filter

1	0	-1
1	0	-1
1	0	-1

take filter and do element wise product. move it along x & y axis

horizontal edge : filter

1	1	1
0	0	0
-1	-1	-1

Sobel filter :

1	0	-1
2	0	-2
1	0	-1

i) more robust
ii) more weight on center

scharr filter :

3	0	-3
10	0	-10
3	0	-3

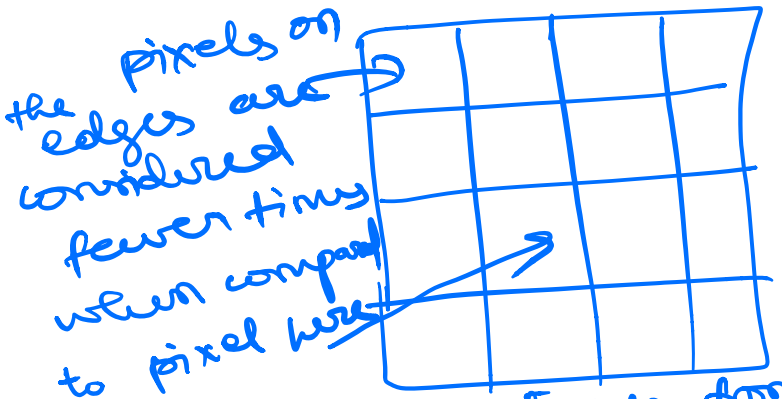
Padding:

input : $n \times n$

filter : $f \times f$

output : $n-f+1,$ $n-f+1$

① Image shrinks



② we throw features from image's edge.

∴ we can pad.

$6 \times 6 \rightarrow 8 \times 8$

filter 3×3 → output : 6×6

out : $n+2p-f+1, n+2p-f+1$

↓

padding size

Valid convolution: no padding

$n \times n \; * \; f \times f \Rightarrow n-f+1 \times n-f+1$

(* : convolve)

Same convolution: pad so that output size == input size

$n + 2p - f + 1 = n$

$2p = f - 1$

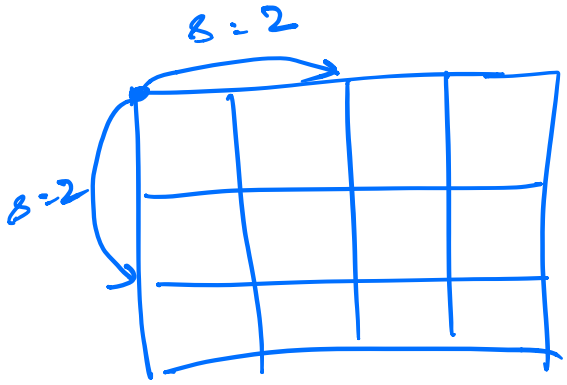
$p = \frac{f-1}{2}$ ⇒ ensures output size same as input size

∴ maintain size of filter as odd

another reason to have f as odd is: we have a center for the filter as one point

Stride:

step by 's' steps instead of 1.



$n \times n * f \times f$

padding p

stride s

$$\text{output} = \frac{n+2p-f}{s} + 1$$

round it down

filter should fully be inside image.
If not dont perform operation.

Usual convolution:

flip filter one time along vertical and horizontal axis.

If we dont flip, its called cross-correlation

Convolutions over volume:

filter has third dimension

h x w x # channel ← input

h_f x w_f x # channel ← filter

the # channels in i/p & filter should match.

$n \times n \times c \quad * \quad f \times f \times c \rightarrow n-f+1 \times n-f+1 \times 1$

we can have multiple filters so:

$n \times n \times c \quad * \quad x\,(f \times f \times c) \rightarrow n-f+1 \times n-f+1 \times x$

↑ no. of filters

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1,1,1,n_c^{[l]})$ ← #filters in layer l.

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$ ←

Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ ←

$$n_{H,W}^{[l]} = \left\lfloor \frac{n_{H,W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$n_c \times n_H \times n_W$

Pooling layer:

parameters: f, s, p → usually zero

max pool: take max of input covered by filter

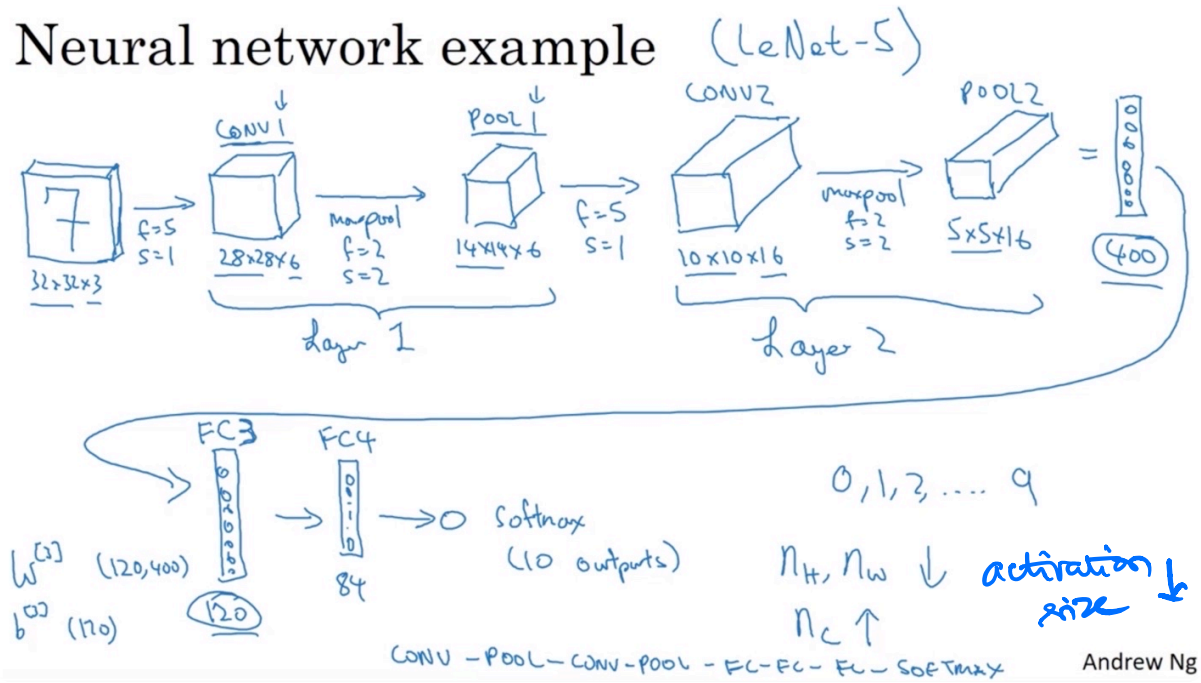
Average: take average

It has no parameters to learn

n_H, n_W, n_c maxpool f, s → $\frac{n+2p-f}{s}+1$, $\frac{n+2p-f}{s}+1$, n_c

for 3D: compute max pool on each channels independently. → same as input

layers = # layers with weight
(so don't consider pool layer)



Why convolution:

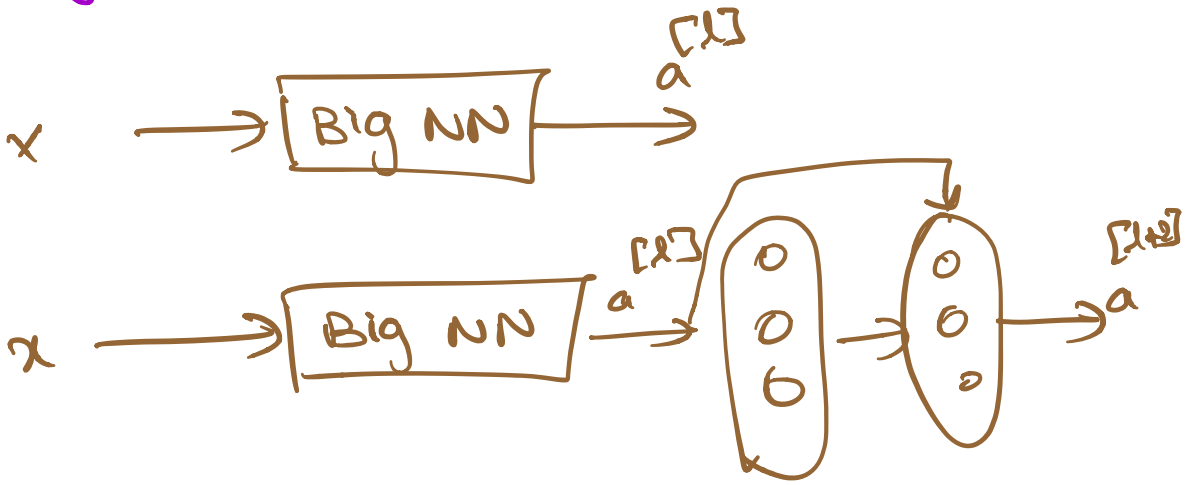
→ helps in parameter sharing

→ sparse connections
(output value depends on small no. of inputs)

Advantages of having a pooling layer:

1) reduce size of representation
2) speed the computation
3) make detected features robust by maintaing the max.

Why residual networks - works?



we use relu

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$$= g(w^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

even if w and $b = 0$, then

$$a^{[l+2]} = g(a^{[l]}) = a^{[l]}$$

∴ even adding 2 extra layers, there won't be any issue / reduction in learning as the last layer will learn identity fn. Therfore, it won't affect the network's performance.

And, as $a^{[l+2]}$ depends on $z^{[l+2]}$ and $a^{[l]}$, we will mostly use "SAME" padding in residual connections so that the shapes remains the same

If size not same

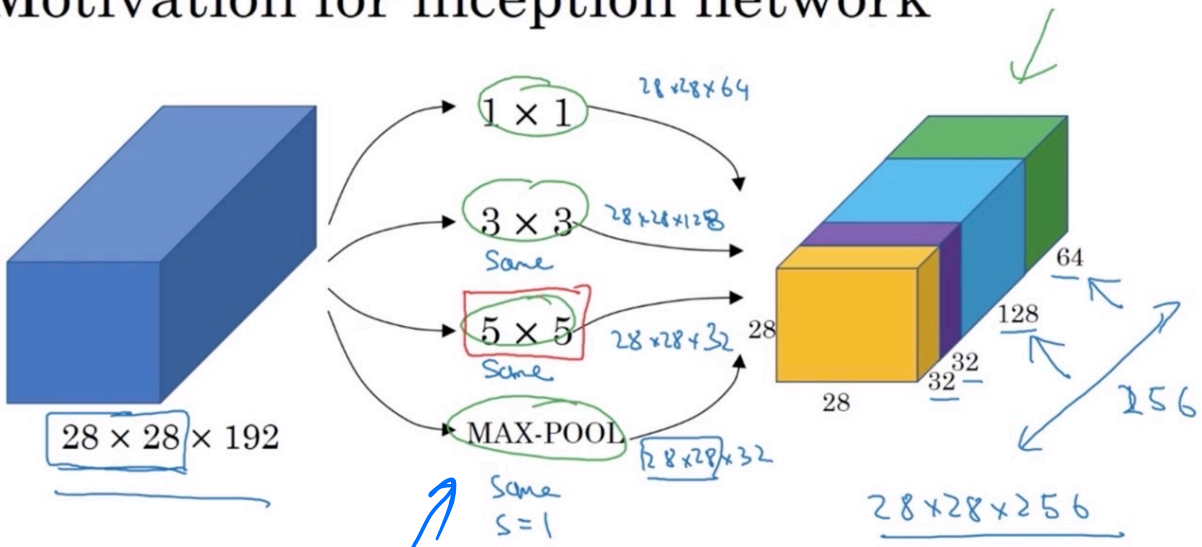
$$\Rightarrow a^{[l+2]} = g\left(z^{[l+2]} + w_s a^{[l]}\right)$$

(w_s: learned or fixed (zero pad))

Network in network (1x1 convo):

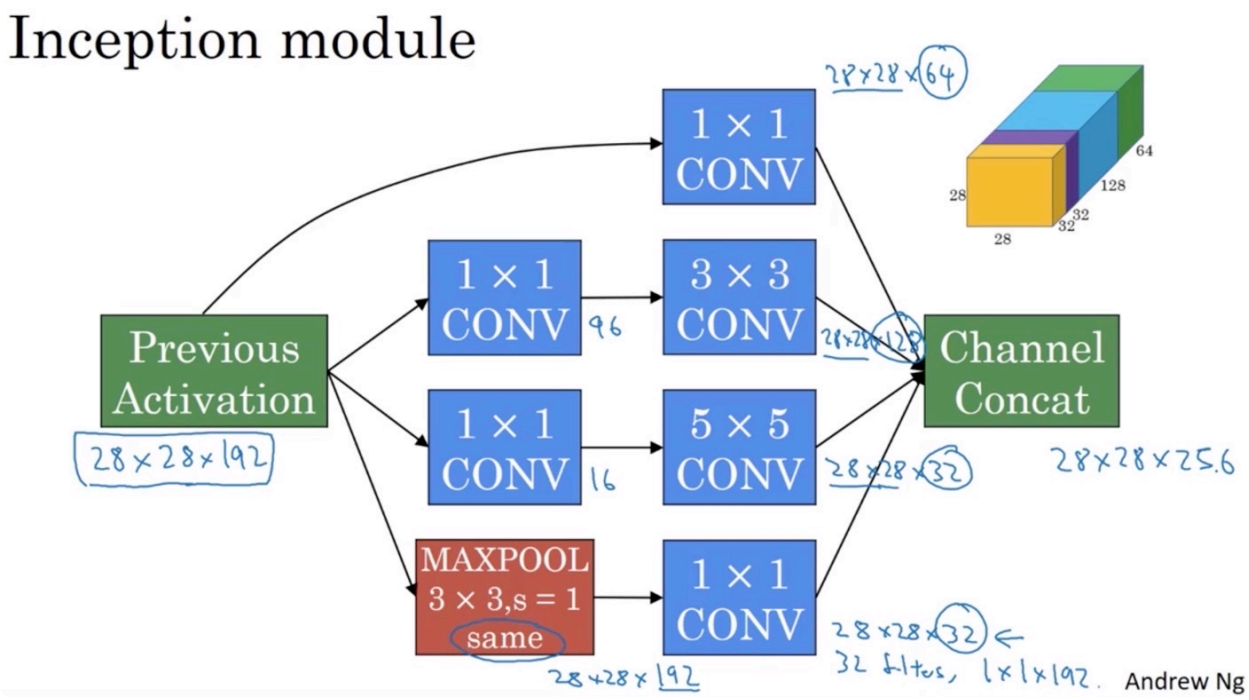
acts as a single neuron that takes all inputs x weights and applies relu.

Motivation for inception network



instead of us selecting what to do, do everything and stack it Algo decides what it wants

To overcome computation cost, use a 1×1 conv to reduce the input size and apply other conv layer on reduced input. This decreases # computat by a large amount.



Data augmentation:

i) flip image
ii) shearing
iii) random cropping
iv) color shifting

Localization:

Tell where the object is

detection:

multiple objects of different category

localization:

make the conv net output 4 more values. (b_x, b_y, b_h, b_w)

b_x, b_y → midpt of bounding box; b_h → height; b_w → width

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

p_c: probability that an object is present (not background)

c_1, c_2, c_3: classes in the image

is there any object (other than background)

there are 8 elements

$$L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots \\ \qquad + (\hat{y}_8 - y_8)^2 \\ \qquad \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 \quad \text{if } y_1 = 0 \end{cases}$$

In practice, we can use:

i) log like feature loss for $c_1 \ldots c_n$ to the softmax output

ii) squared error for bounding box

iii) logistic regression loss for P_c

we can also make a CNN to output landmark / exact x, y point in an image

Object detection:

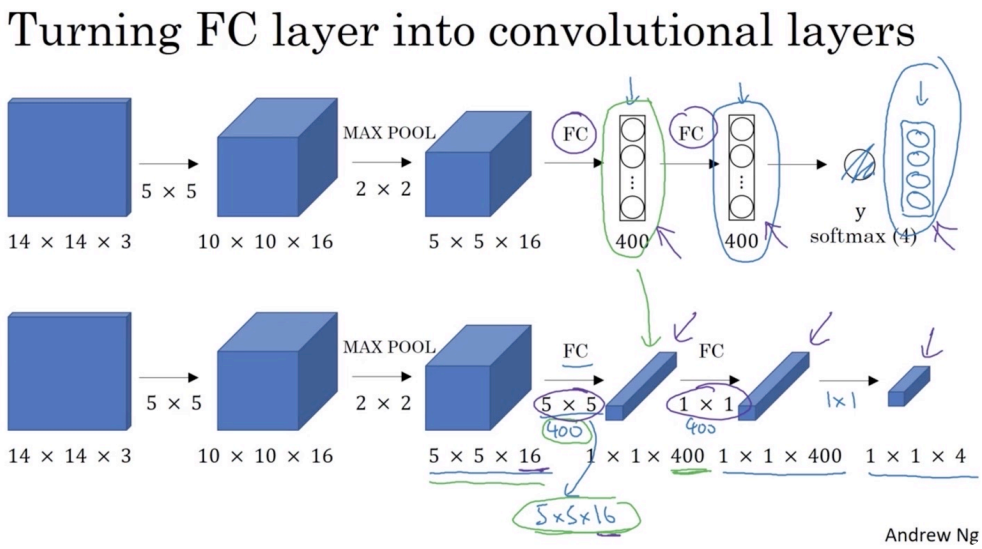
i) train a convnet on closely cropped images of the object.

ii) Use sliding window on the input image and detect object in it using above convnet.

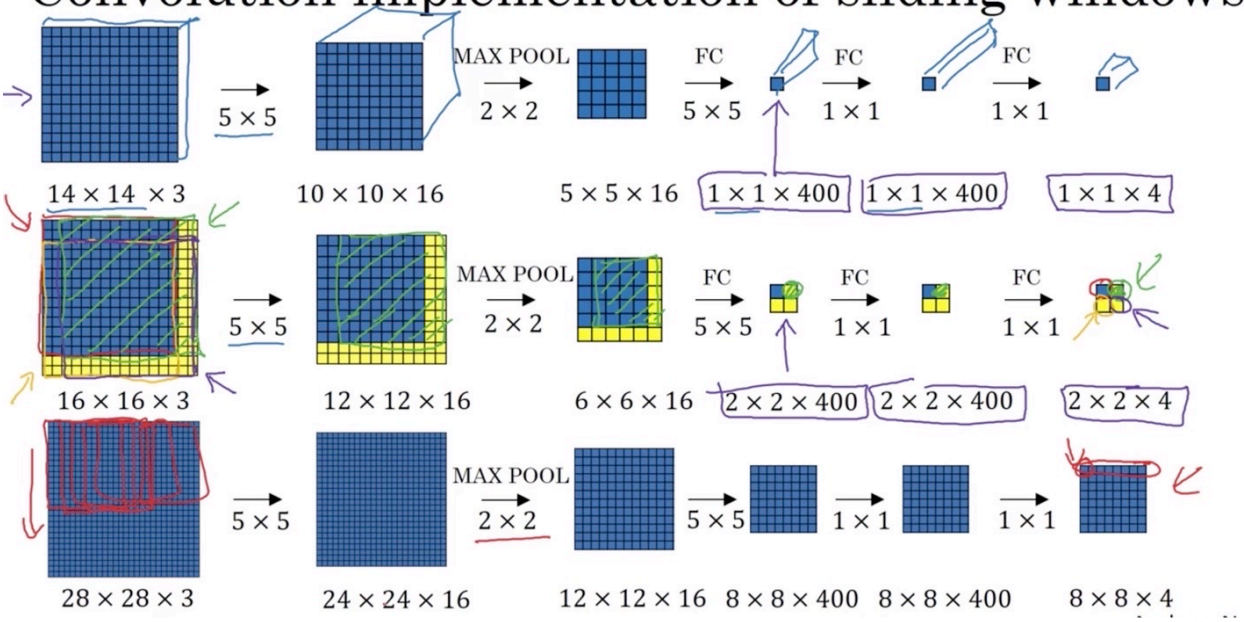
do this repeatedly for varying window size

This is called sliding window detection

Computational cost is very high



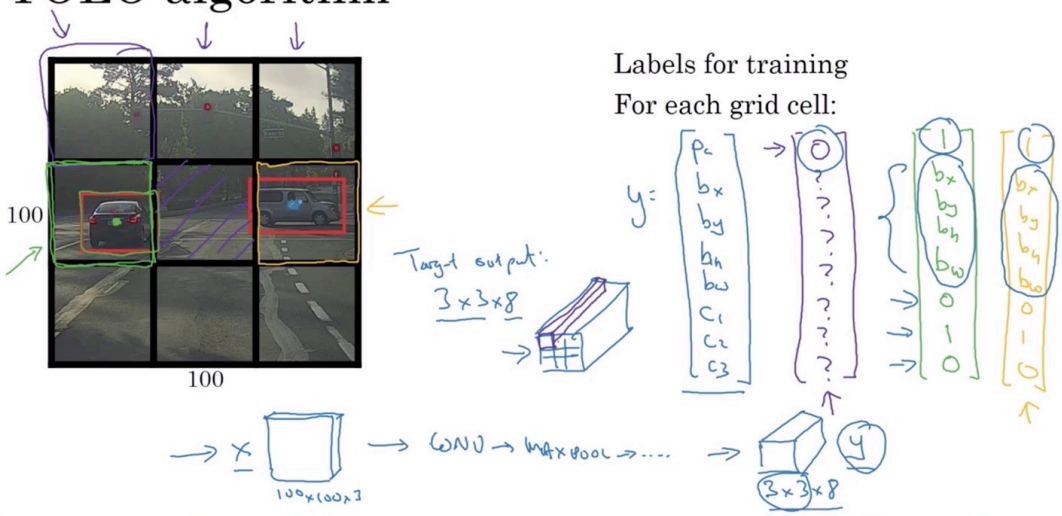
Convolution implementation of sliding windows



Still the predictions are not so accurate!

You only look once algo: (YOLO)

YOLO algorithm



Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

Summary

i) split image into grids

ii) use object localization in each grid.

This can be done using a fully conv net and output $n \times n \times 8$ ← corresponds to 8 value in obj. local output.

↑ no. of grids

here in Yolo, each grid's starting is (0,0) and end is (1,1)

so b_x, b_y, b_h, b_w are relative to grid cell

Intersection over union:

$$= \frac{\text{size of intersection}}{\text{size of union}}$$

"correct" if $IOU \geq 0.5$

Non max suppression:

Get 19x19x8 output after running object detection.

consider each bounding box has a probability / confidence. Discard $P_c \leq 0.6$

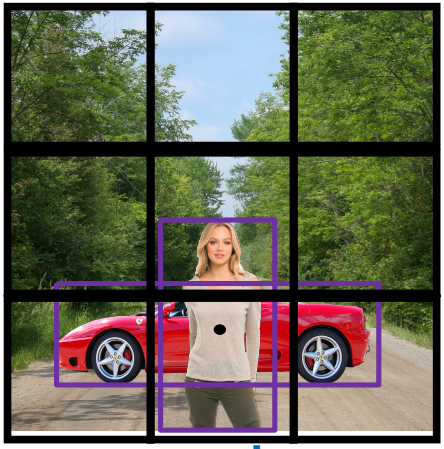
Take the max prob.

Find other boxes that has high IOU (≥ 0.5) with this box and suppress it.

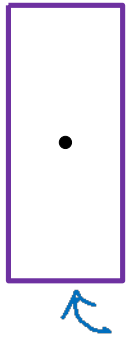
continue until we have non overlap box. (i.e) until we have some remaining boxes.

If we have multiple objects, do non-max suppression independently on each box.

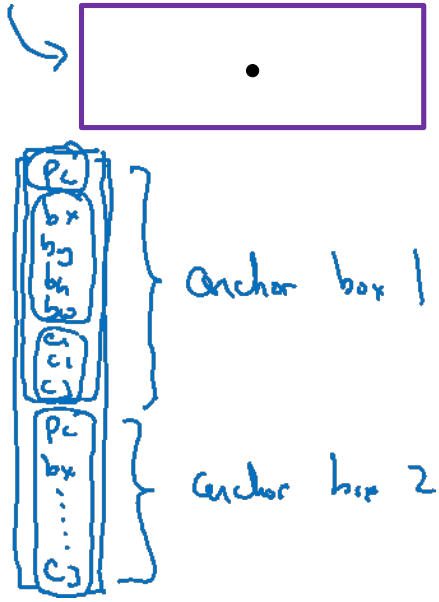
Overlapping objects:

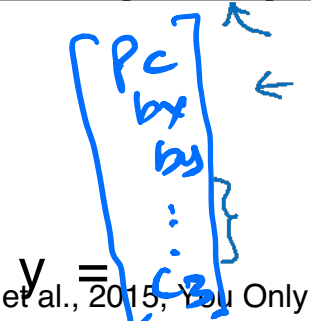


Anchor box 1:



Anchor box 2:





[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

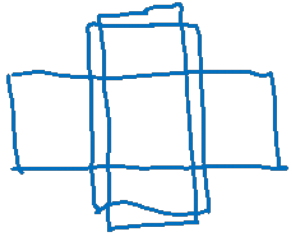
Anchor box algorithm

Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

Output y:

$3 \times 3 \times 8$



With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

(grid cell, anchor box)

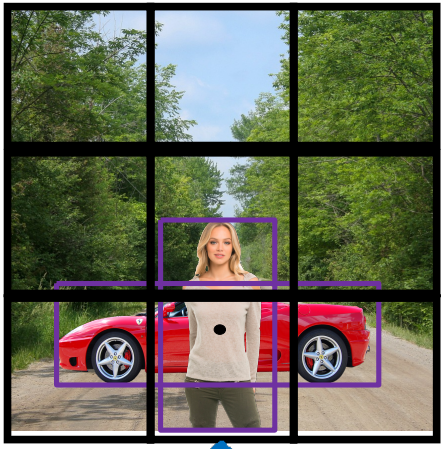
Output y:

$3 \times 3 \times 16$

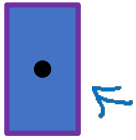
$3 \times 3 \times 2 \times 8$

Anchor box example

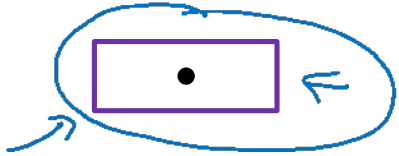
y =

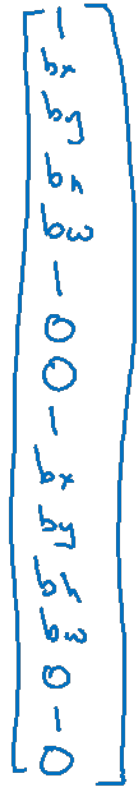


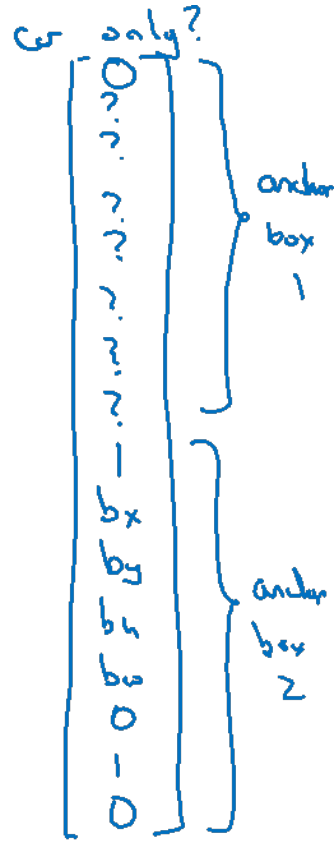
Anchor box 1:



Anchor box 2:







RCNN:

instead of using sliding window on all possible window, select regions on which to run convnet classifier.

First run segmentation algo to get segmented image and use convnet on a blob in segmented image to get label and bounding box.

This method is slow

Fast-R-CNN: use convolutional sliding window.

Faster-R-CNN: use CNN for segmentation

Face recognition:

face verification vs recognition

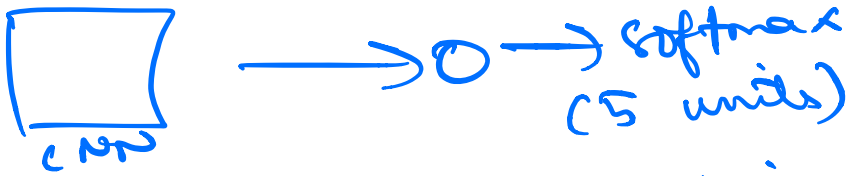
input image, name/ID
output: if input image is that of claimed person
↓
1:1

* has a dataset of K person
* input image
* output ID if image is any of K person
↓
1:K

one shot learning:

learn from one example to recognize the person again

general approach: (4 persons)



but what if new person joins?
⇒ we have to train whole network with 6 o/p units

So, learn similarity fn:

$d(img1, img2)$: degree of diff. between images

during test:

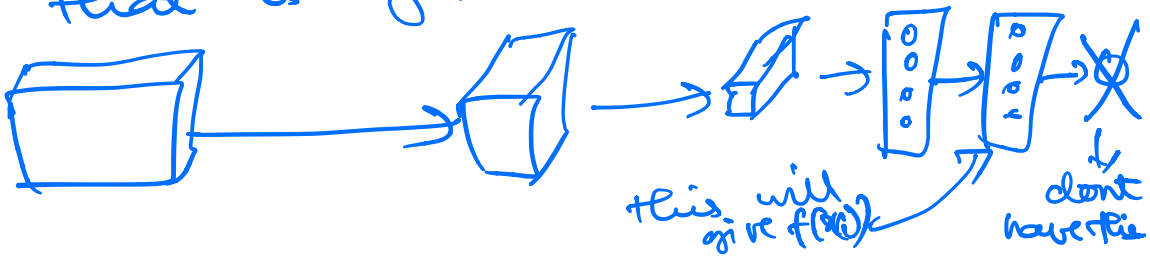
if $d(img1, img2) \leq \tau \rightarrow$ same person

$> \tau \rightarrow$ diff person

compute $d(img, img2)$ $\forall$ img2 $\in$ dataset and see id where it is small.

Siamese network:

train a NN with last layer giving the encoding of image that is given as input.



Learn parameters such that:

If $x^{(i)}$, $x^{(j)}$ are same:

$$\|f(x^{(i)}) - f(x^{(j)})\|_2^2$$

is small

if $x^{(i)}$, $x^{(j)}$ are different:

$$\|f(x^{(i)}) - f(x^{(j)})\|_2^2$$

is large

Triplet loss:

we have an anchor image and a positive image.

⇒ distance has to be small

anchor image and a negative image

⇒ distance has to be large

Triplet loss: look 3 images Anchor, positive, negative

A P N

want : $\|f(A) - f(P)\|_2^2$

$\leq \|f(A) - f(N)\|^2$

$d(A,P) \leq d(A,N)$

$\|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|^2 \leq 0$ $-\alpha$

$\|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|^2 + \alpha \leq 0$

this margin is added so that our CNN doesn't output 0 for all encodings

loss fn:

given : A, P, N (3 images)

$L(A,P,N) = \max(\|f(A)-f(P)\|_2^2 + \|f(A)-f(N)\|_2^2 + \alpha, 0)$

we take max because term A should be ≤ 0. If it is ≤ 0, loss is 0

Since we need A, P to be same person, for training we need multiple pictures of same person But while testing, we need only one image per person

If A, P, N are chosen randomly (A ≠ P) then it will be easy to satisfy triplet loss fn and our NN wont learn anything.

∴ we need to choose triplets that's hard to train on

$$d(A,P) + \alpha \leq d(A,N)$$

$$d(A,P) \approx d(A,N)$$

⇒ learning becomes hard

we can also treat this as a binary classification where we feed $f(x^{(i)})$ and $f(x^{(j)})$ to a logistic regression and output 1 (same) or 0 (diff. person)

$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b\right)$$

$$\chi^2_{\text{(chi square)}} = \frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$$

Neural Style transfer:

content (C) + style (S) → generated image (G)

$J(G) = \alpha J_{content}(C, G)$

find G: $+\beta J_{style}(S, G)$

i) randomly initiate G

ii) use gradient descent to minimize $J(G)$

$G: G - \frac{\partial}{\partial G} J(G)$

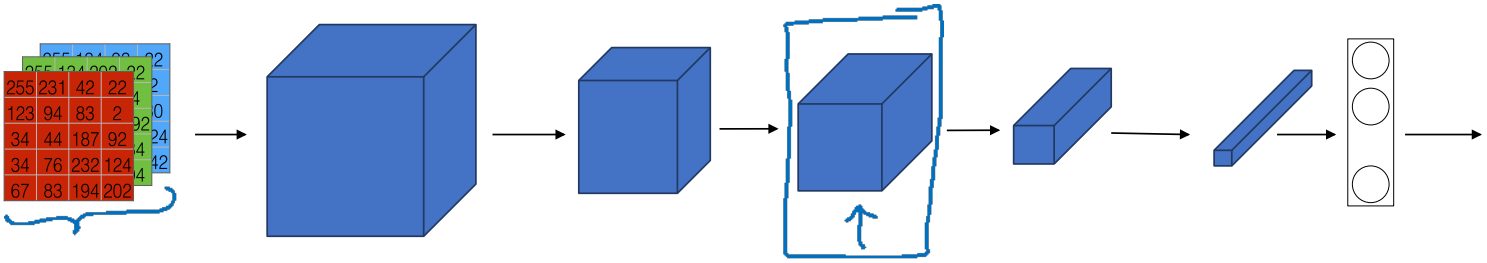
Content cost function

$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

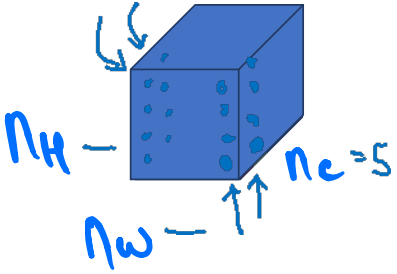
- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

$$J_{content}(C, G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

Meaning of the “style” of an image



Say you are using layer ’s activation to measure “style.”
Define style as correlation between activations across channels.

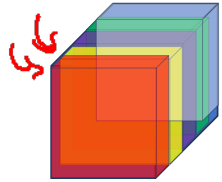


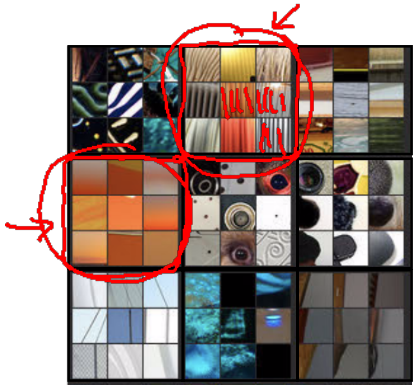
How correlated are the activations across different channels?

[Gatys et al., 2015. A neural algorithm of artistic style]

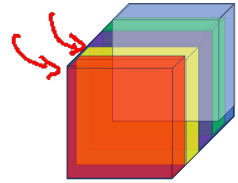
Intuition about style of an image

Style image





Generated Image



Correlated?
Uncorrelated

Components appears together

So in generate image's activation, we can find correlated styles and use it.

[Gatys et al., 2015. A neural algorithm of artistic style]

Style matrix:

Let $a^{[l]}_{i,j,k}$ = activation at (i,j,k) (H, W, C)

$G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

↑ each pair of channels' correlation

gram matrix → $G^{[l]}_{kk'}$ → correlation of channel k and k'

$$G^{[l]}_{kk'} = \sum_{i}^{n_H^{[l]}} \sum_{j}^{n_W^{[l]}} a^{[l]}_{ijk}\, a^{[l]}_{ijk'}$$

↑ iterate over 0 to n_c

Do $G^{[l]}_{kk'}$ for both style and generated image

$$J^{[l]}_{style}(S,G) = \| G^{[l](S)} - G^{[l](G)} \|_F^2$$

$$= \frac{1}{\left(2\, n_W^{[l]} n_H^{[l]} n_c^{[l]}\right)^2} \sum_{k} \sum_{k'} \left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'} \right)^2$$

$$J_{style}(S,G) = \sum_{l} \lambda^{[l]} J_{style}^{[l]}(S,G)$$

↑ iterate over diff layers for getting pleasing result